基于节点影响力和权重聚合签名的改进 PBFT 共识算法

刘力汇[1], 邓小鸿[2,3], 刘 勇[1], 石亦燃[1], 张 丽[1]

[1]江西理工大学 信息工程学院 赣州 中国 341000
[2]赣南科技学院 信息工程学院 赣州 中国 341000
[3]赣州市云计算与大数据重点实验室 赣州 中国 341000

摘要 共识算法是区块链中确保数据达成一致的关键技术。实用拜占庭容错(Practical Byzantine Fault Tolerance, PBFT)共识算法能够有效解决拜占庭将军问题, 凭借出色的容错性和高效性, 被广泛应用于分布式系统、区块链等场景。但是 PBFT 共识算法存在着缺少奖惩机制、主节点选取方式可预测和通信开销大等问题, 针对上述问题, 本文提出了一种基于节点影响力和权重聚合签名的改进 PBFT 共识算法。首先, 设计了信誉模型动态选取共识节点, 依据节点类型给予不同程度奖惩, 并设计了信誉恢复机制防止“寡头”节点产生; 其次, 建立影响力评估机制, 提出一种新的结合整体和局部结构的 K-Shell 算法对共识节点进行影响力评估, 识别共识网络中的关键节点。同时设计了基于节点影响力的可验证随机函数, 在提高关键节点成为主节点概率的同时, 使选取方式具有不可预测性; 最后, 提出权重聚合签名方案优化共识流程, 通过赋予节点权重进行聚合签名, 降低了共识过程的通信开销和签名量, 提升了算法的共识效率。实验结果表明, 与原始 PBFT 相比, 本文算法的平均吞吐量提高了 65.7%, 平均时延降低了 38.9%, 有效地提高了系统共识效率, 另外, 与典型的 PBFT 改进算法相比, 本文算法具有明显的性能优势, 能更好地适用于大规模联盟链场景。

关键词 区块链; 共识算法; 节点影响力; K-Shell 算法; 聚合签名
中图法分类号 TP309 DOI 号 10.19363/J.cnki.cn10-1380/tn.2026.03.10

Improved PBFT Consensus Algorithm Based on Node Influence and Weighted Aggregation Signature

LIU Lihui[1], DENG Xiaohong[2,3], LIU Yong[1], SHI Yiran[1], ZHANG Li[1]

[1] School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China
[2] School of Information Engineering, Gannan University of Science and Technology, Ganzhou 341000, China
[3] Key Laboratory of Cloud Computing and Big Data, Ganzhou 341000, China

Abstract The consensus algorithm is the key technology in blockchain to ensure that data is agreed upon. Practical Byzantine Fault Tolerance (PBFT) consensus algorithm can effectively solve the Byzantine Generals problem, and is widely used in distributed systems, blockchain and other scenarios due to its excellent fault tolerance and high efficiency. However, the PBFT consensus algorithm has problems such as lack of reward and punishment mechanism, predictable master node selection and high communication overhead, etc. Aiming at the above-mentioned issues, we proposed an improved PBFT consensus algorithm based on the influence of nodes and weighted aggregation signature. Firstly, a reputation model is designed to dynamically select consensus nodes, with different levels of rewards and penalties based on node types, and a reputation recovery mechanism is designed to prevent “oligopoly” nodes from being generated. Secondly, established the influence assessment mechanism, proposed a novel K-Shell algorithm combining the global and local structure to assess the influence of consensus nodes and identify the key nodes in the consensus network. Meanwhile, a verifiable random function based on the influence of nodes is designed, which improves the probability of key nodes becoming master nodes while making the selection method unpredictable. Finally, we proposed the weight aggregation signature scheme to optimize the consensus process, which reduces the communication overhead and signature volume of the consensus process and improves the consensus efficiency of the algorithm by assigning weights to the nodes for the aggregation signature. Experimental results show that compared with the original PBFT, the average throughput of this paper’s algorithm is improved by 65.7%, and the average delay is reduced by 38.9%, which effectively improves the consensus efficiency of the system. In addition, compared with the typical improved algorithm of PBFT, this paper's algorithm has an obvious performance advantage, and it can be better applied to large-scale consortium chain scenarios.

通讯作者: 邓小鸿, 博士, 副教授, Email: deng_xh@jxust.edu.cn。
本课题得到国家自然科学基金(No. 61762046, No. 62166019), 江西省自然科学基金(No. 20224BAB202019)资助。
收稿日期: 2024-06-25; 修改日期: 2024-08-23; 定稿日期: 2026-01-26

Key words blockchain; consensus algorithm; node influence; K-Shell algorithm; aggregate signature

1 引言

比特币[1]问世以来, 其底层核心的区块链技术引起了研究者的广泛关注。区块链技术具有去中心化、数据难篡改、可追溯等特点, 早期在数字货币领域得到广泛应用。如今, 区块链技术经过不断的演变发展, 其应用场景越来越广泛, 不再局限于数字货币领域, 已应用于能源[2]、物联网[3]、医疗[4]等领域。共识算法作为区块链的核心组成部分, 决定着数据以何种方式达成一致, 直接决定了区块链系统的性能, 共识算法的优化研究已经成为当前研究热点。

目前常见的共识算法有工作量证明(Proof of Work, PoW)[5]、权益证明(Proof of Stake, PoS)[6]、委托权益证明(Delegated Proof of Stake, DPoS)[7]、Raft(Replication and Fault Tolerant)[8]、实用拜占庭容错[9]等。PoW 共识算法在公有链中应用广泛, 但效率较低同时浪费大量的算力资源。PoS 共识算法针对 PoW 存在的问题引入币龄的概念, 但去中心化程度较低, 记账权被少部分币龄大的节点掌握。DPoS 是 PoS 衍生算法, 通过投票方式选举出部分节点作为记账节点轮流生成区块, 虽然会产生节点不积极投票的问题, 但相比于 PoW 和 PoS, 能够降低能耗, 提升共识效率。Raft 共识算法因其通信复杂度低在私有链中被广泛应用, 但不能容忍拜占庭故障。相比之下, PBFT 共识算法不依赖代币, 共识过程无资源浪费和分叉情况, 同时能够容忍拜占庭节点, 因此被广泛应用于联盟链中。尽管如此, PBFT 算法仍然有以下不足之处: (1) 缺少奖惩机制, 恶意节点将持续存在系统中参与共识, 存在安全隐患, 同时诚实节点得不到激励, 降低了参与共识的积极性。(2) 主节点选取随意, 所有节点轮流担任存在可预测问题, 同时影响了主节点的可靠性, 恶意节点成为主节点将导致频繁的视图切换降低共识效率。(3) 共识流程通信复杂度高, 当节点数量增加时, 需要较大的通信开销才能达成一致。

针对 PBFT 共识算法存在的问题, 研究人员从不同角度提供了优化方法, 目前的研究工作可以划分为以下三类方向。

方向 1: 减少参与共识的节点数量。Gan 等人[10]通过聚类算法对网络中的节点进行分组, 并使用基于位置和信誉的评分机制, 通过计算节点的地理值和信任值筛选出共识节点, 进一步减少了各分组内节点的数量, 提高了组内共识性能。Liu 等人[11]提出一种基于信用分组的高效 PBFT 共识算法, 通过引入优化的三路快速排序算法将节点划分到具有不同功能的三个分组中, 减少了参与共识的节点数量, 提高了分组和共识效率, 并设计了组间身份转换机制, 及时替换信誉较低的共识节点。Liu 等人[12]利用 CART 决策树算法对节点进行评估和分类, 并通过信誉模型将节点划分为共识节点、候选节点和备份节点三种类型, 依据节点行为动态调整节点类型, 提升了共识节点的可靠性。以上几种算法都考虑使用节点分组的方式减少参与共识的节点数量, 一定程度上降低了通信开销和提升了共识效率, 但忽略了复杂分组方式带来的额外开销问题。例如文献[10]采用聚类算法分组, 在节点数量较多的情况下聚类算法需要较高的计算开销, 且动态性较低, 节点退出或加入需要频繁重新聚类。同时未考虑进一步细化奖惩机制, 忽略对不同类型节点功能差异的信誉奖惩问题, 且上述研究的信誉模型对高信誉节点未作处理, 存在信誉“寡头”节点问题, 高信誉节点将长期参与共识, 降低了低信誉节点的共识积极性。

方向 2: 优化主节点选取方式。Na 等人[13]提出一种双主节点共识算法, 在 PBFT 主节点选取的基础上随机从全体节点中再选出另一个节点作为副主节点, 两个主节点一个负责转发请求消息, 一个负责收集并验证消息, 通过双主节点间相互监督合作, 提升了系统性能。Wang 等人[14]通过特征信任模型对节点进行信任度评估, 按照信任度大小选取主节点, 提升了主节点的可靠性。Tang 等人[15]提出一种针对联盟链高频交易场景的改进 PBFT 算法, 在主节点选取上使用可验证随机函数(Verifiable Random Function, VRF)[16]保障了主节点选取的不可预测性, 提升了系统安全性。上述工作通过信誉选取或随机选取的方式提高了主节点的可靠性, 保障了选举过程的安全性。但是选取信誉值最高节点担任主节点的方式仍无法解决可预测问题, 并且仅通过信誉属性选取主节点具有片面性, 信誉高的节点并不一定能够承担主节点的任务, 而随机选取方法缺少门槛限制容易选择不稳定和恶意节点, 同时均匀的随机选取不能体现出对不同权重大小节点的公平性。

方向 3: 优化共识流程。Li 等人[17]提出一种基于完美二叉树通信拓扑结构的拜占庭容错共识算法, 依据信誉模型构建完美二叉树通信拓扑结构, 在共识流程中加入流水线机制将上一轮共识中的确认阶段与下一轮共识中的准备阶段同时进行, 提高了共

识效率。Jiang 等人[18]根据树状拓扑结构将节点划分为不同的层和组, 共识先由父节点达成再向下传递给子节点, 实现了从全局共识到局部共识的转换, 降低了通信开销。上述方法通过对拓扑结构进行改进优化了共识流程, 有效降低了通信开销, 但未考虑进一步减少共识过程中节点签名和验证签名的次数。例如文献[17]虽然在共识过程中采用了聚合签名, 但相同权重的聚合签名无法体现出不同节点的重要性, 越靠近二叉树根部的节点其重要程度显然更高。而文献[18]仅通过划分节点层次的方式减少签名量, 其共识流程仍然需要大量的签名和验证签名次数。

针对上述研究工作存在的问题, 本文提出一种基于节点影响力和权重聚合签名的改进 PBFT 共识算法(Node Influence and Weighted Aggregation Signature PBFT, IW-PBFT)。本文的主要贡献如下:

(1) 提出分级奖惩和抗“寡头”的动态信誉模型。通过信誉排序动态选取共识节点, 并进一步细化奖惩机制, 根据节点类型和行为进行分级奖惩。在信誉模型中加入惩罚因子加快对恶意节点的剔除, 提高了参与共识的节点质量, 同时设计了信誉恢复机制, 避免“寡头”节点的产生。

(2) 提出基于节点影响力的主节点选择方法, 包含两部分创新: 改进 K-Shell 算法的节点影响力评估机制和基于节点影响力的可验证随机函数(Verifiable Random Function Based on Node Influence, I-VRF)。在信誉模型筛选出共识节点的基础上, 设计结合整体和局部结构的 K-Shell 算法(Combining Global and Local Structures K-Shell Algorithm, GLK-Shell)对共识节点进行影响力评估, 识别共识网络中的关键节点。相比于 K-Shell 算法, GLK-Shell 算法引入了迭代次数和一二阶邻居因素, 具有更好的影响力识别能力, 提升了主节点选取属性的综合性。I-VRF 在保障主节点选择随机性的同时, 使得节点被抽取概率与自身影响力大小相关, 实现了对不同权重大小节点的公平抽取, 提升了关键节点当选主节点的概率, 有效解决了主节点可预测的问题。

(3) 设计基于节点权重的聚合签名方案。通过影响力排序结果为共识节点分配权重, 并依据权重是否达到阈值进行聚合签名, 与聚合签名方案相比有更低的聚合门槛, 减少了共识流程中的通信开销和签名量, 提高了算法共识效率。

2 问题的提出

2.1 PBFT 算法及其存在的问题

PBFT 是一种基于投票的共识协议, 每个视图中仅有一个主节点领导共识进行, 所有节点都将参与投票, 共同确保消息的一致性。假设节点总数为 N, 当前视图编号为 v, 则主节点编号为 $p=v \bmod N$, 系统能够容忍最多 f 个拜占庭节点, 仅当 f 与 N 满足 $3f+1 \le N$ 时, 算法能够正常进行共识。

在开始共识前, 需在当前视图中选择一个节点作为主节点。PBFT 主节点选择如图 1 所示, 其中节点 1、2、3 为正常节点, 节点 4 为恶意节点。在初始视图 $v=1$ 时, 主节点编号为 $p=1 \bmod 4=1$, 节点 1 成为主节点, 当视图编号发生变化, 视图 $v=2$ 时, 主节点编号为 $p=2 \bmod 4=2$, 节点 2 成为主节点, 依此类推, 随着视图编号 v 的增加, 各节点将轮流担任主节点。

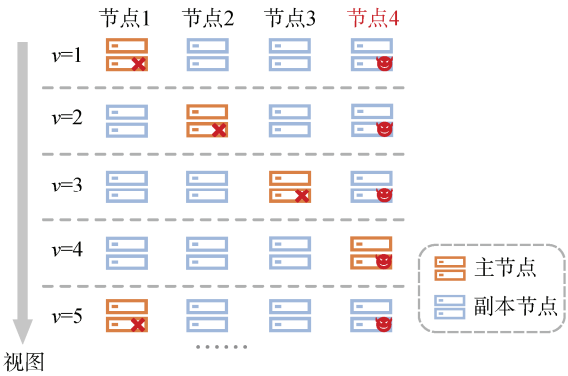


图 1 PBFT 主节点选择

Figure 1 PBFT primary node selection

从图 1 中可以看出 PBFT 主节点选择存在以下问题: (1) 所有节点均可担任主节点, 缺少门槛限制, 同时缺乏恶意节点惩罚机制。例如图 1 中恶意节点 4 也能够成为主节点, 并且将持续存在系统中, 在下一轮循环中将继续担任。(2) 节点轮流担任主节点存在可预测风险。例如图 1 中, 假设视图 $v=1$ 时对手攻击节点 1, 抑制其发送预准备消息, 其他节点在规定时间内未收到主节点消息发起视图更换, 此时视图 $v=2$, 按照轮流担任规则节点 2 成为主节点。同时, 对手根据轮流担任规则预测了节点 2 为新主节点, 再次发起攻击, 其他节点未收到新主节点消息, 再次发起视图更换。依此类推, 对手很容易预测下一主节点进行攻击从而破坏共识。

PBFT 共识流程如图 2 所示, 共包含 5 个阶段, 设节点 0 为主节点, 节点 1、2、3 为副本节点, 其中节点 3 为拜占庭节点, 将在准备和确认阶段干扰其他正常节点共识。从图 2 中可以看出 PBFT 共识流程存在以下问题。

(1) 恶意节点持续影响共识。在准备阶段中, 拜占庭节点 3 向主节点 0 发送同意消息, 认为请求是合

法有效的, 但向节点 1 和 2 发送反对消息, 认为请求是非法无效的。此时节点 1 在收到拜占庭节点 3 发送的消息后无法判断请求是否合法, 无法进入下一阶段, 只有当接收到来自节点 2 的消息后才能进行判断。同理, 在确认阶段中, 拜占庭节点 3 仍然通过这种选择性发送消息的方式干扰正常节点共识。凭借 PBFT 自身的容错能力, 仍然能够在拜占庭节点数量较少时对请求达成一致, 但拜占庭节点将持续存在系统中干扰共识, 若拜占庭节点数量超过最大容忍量, 算法将无法确保一致性。

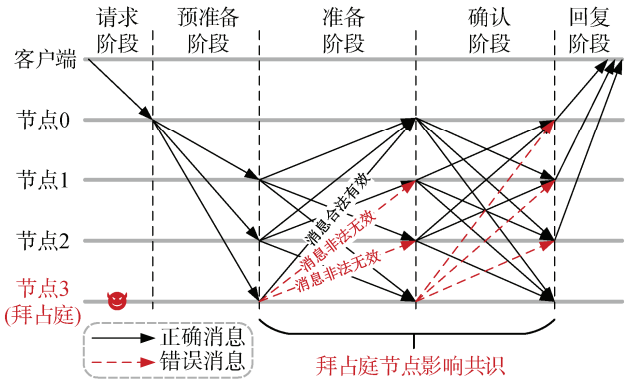


图 2 PBFT 共识流程

Figure 2 PBFT consensus process

(2) 需要大量的通信开销和签名量。例如, 在确认阶段中主节点 0 需要与节点 1、2、3 进行通信, 每次发送消息都需要进行一次签名, 节点接收到每条消息都需要对签名进行验证。假设此时系统中共有 N 个节点, 则确认阶段所需的通信总次数为 N^2-N, 签名和验证签名总次数为 $2N^2-2N$, 而完整的 PBFT 共识流程需要更大的通信开销和签名量, 将导致共识效率随着节点规模增大而快速下降。

除此之外, 恶意节点在共识中的安全隐患还包括: (1) 恶意节点试图通过贿赂主节点发布非法交易, 或向其他共识节点提供利益的方式获得它们的支持, 从而得到记账权。(2) 恶意节点试图通过控制目标节点的网络连接, 使其无法接收到正常节点的消息或只能接收到恶意节点的消息, 造成网络分区的现象。(3) 恶意节点试图通过伪装成多个虚假身份的方式欺骗其他节点, 以获得更多的投票权从而控制共识。

2.2 K-Shell 算法及其存在的问题

对复杂网络中的节点进行重要性排序能够为评估节点潜在风险、加强关键节点安全防护, 以及确定节点风险处置优先级顺序提供有力的依据。K-Shell 算法是评估网络中关键节点的常用算法, 一般认为, 单个节点与其他节点联系次数越多, 越靠近网络中心位置, 则成为核心节点的概率越大, 其网络影响力也越大。基于这一思想, Kitsak 等人[19]于 2010 年开创性提出 K-Shell 分解法, 通过分析网络节点的度数和连接关系, 区分网络节点重要性和影响力。

K-Shell 的分解过程如图 3 所示, 具体过程如下: 首先, 剔除所有度为 1 的节点以及与它相连的边, 如图 3 中的节点 1 和它与节点 3 相连的边。其次, 此时网络中会出现新的度为 1 的节点, 如图 3 中的节点 3 在节点 1 被剔除后其度值变为 1, 继续剔除这些节点, 直至不再出现度为 1 的节点, 将所有被剔除的节点划为 1-Shell 层, 分配 Ks 值为 1。最后, 重复上述过程, 依次迭代删除度为 2, 3, 4, …的节点, 直至每个节点都有唯一对应的 Shell 层和 Ks 值。其中, Ks 值最大的 Shell 层被认为是网络的核心层, 该层节点为网络中的关键节点, 具有最大的影响力。从分解过程可以看出, K-Shell 算法的计算复杂度取决于图的稀疏程度和表示方式。对于有 n 个节点和 e 条边的图, 用邻接矩阵表示时, 每个节点度数的更新和删除都需要遍历所有节点, 使得 K-Shell 算法的计算复杂度为 $O(n^2)$。用邻接表表示时, 上述操作都能在常数时间内完成, 使得 K-Shell 算法的计算复杂度为 $O(n+e)$。表 1 为对图 3 的 K-Shell 分解过程记录。

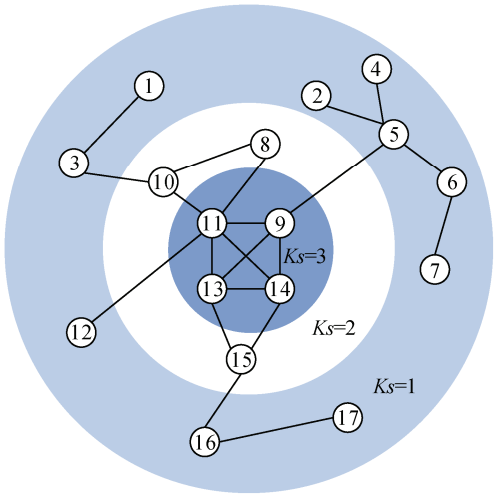


图 3 K-Shell 分解示例图

Figure 3 K-Shell decomposition example

表 1 K-Shell 分解过程

Table 1 K-Shell decomposition process

迭代次数	剔除的节点	Ks 值
1	1, 2, 4, 7, 12, 17	1
2	3, 6, 16	1
3	5	1
4	8, 10, 15	2
5	9, 11, 13, 14	3

虽然 K-Shell 算法能够快速划分节点的影响力层次, 但从图 3 和表 1 中能看出 K-Shell 算法存在以下问题: (1) 划分结果是粗粒度的。同一层的所有节点处于同等地位, 并未进一步区分其影响力, 例如图 3 中, 节点 2 和节点 5 被划分到同一层中, 被分配相同的 *Ks* 值, 但节点 5 度数更大并且是连接不同层次的桥梁, 其影响力显然比节点 2 高。(2) 未考虑迭代次数。在分解过程中, 大量的迭代次数不同的节点被划分到同一层, 例如表 1 中, 由于未考虑迭代次数的影响, 迭代次数分别为 1、2、3 的共 10 个节点均被划分到 1-Shell 层中。

3 IW-PBFT 共识算法

3.1 算法模型

IW-PBFT 算法通过信誉排序的方式, 动态筛选出共识节点, 在共识节点中通过影响力评估抽取主节点, 同时, 将权重聚合签名与共识流程结合, 减少了算法的通信开销和签名量。以系统中 7 个节点为例, 算法整体模型如图 4 所示, 主要分为以下步骤:

(1) 信誉排序。在每一轮共识前, 通过信誉模型按照信誉高低进行排序, 排名前 *N–f* 的节点作为共识节点参与共识, 其余节点作为备份节点同步共识结果。如图中节点 1~5 被划分为共识节点, 节点 6 和 7 为备份节点。

(2) 共识节点影响力评估。共识节点间进行投票, 每个节点拥有一票, 各节点依据其他节点的信誉值、共识行为和与自身的交互历史等因素选择信任的节点进行投票。将节点间的投票通过线段连接, 从而构建全局投票网络图, 使用 GLK-Shell 算法计算各节点的综合影响力。图中共识节点 1~5 进行投票, 之后 GLK-Shell 算法对全局投票图先进行整体结构影响分析, 节点 4 和节点 5 划为 1-Shell 层, 节点 1~3 划为 2-Shell 层, 再分别对各层进行局部结构影响分析, 得到各节点的综合影响力。

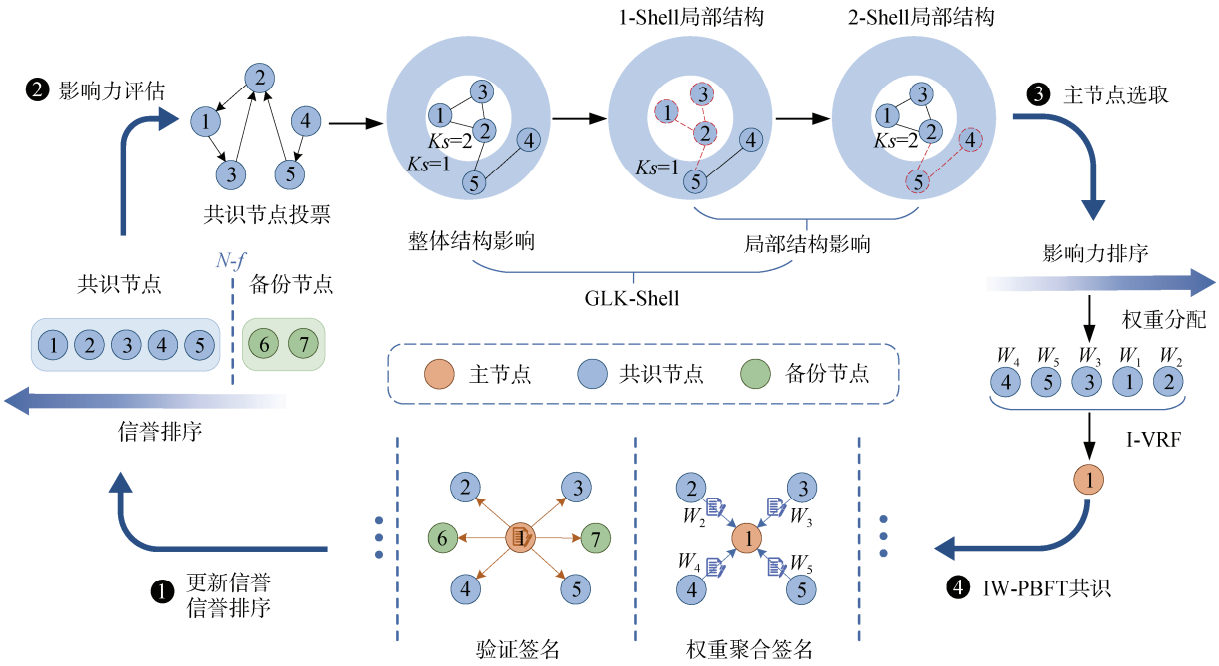


图 4 IW-PBFT 共识算法模型

Figure 4 IW-PBFT consensus algorithm model

(3) 主节点选取。依据影响力大小对共识节点进行排序, 并按照排序结果为各节点分配权重, 为后续共识和主节点选取做准备, 使用 I-VRF 算法进行权重随机抽取主节点。图中共识节点 1~5 按照排序结果获得自身权重, I-VRF 算法将节点 1 抽取为主节点。

(4) 开始共识。共识节点执行 IW-PBFT 共识流程, 在权重聚合阶段, 将通过权重聚合签名方案生成聚合签名, 并将签名发送给共识节点和备份节点验证。在每轮共识完成后, 更新节点信誉开始下一轮共识。图中共识节点 2~5 将附带自身权重的签名发送给主节点 1, 主节点在判断权重达到阈值后生成聚合签名并发送给节点 2~7 进行验证。

3.2 分级奖惩和抗“寡头”的动态信誉模型

为了提升参与共识的节点质量, 降低恶意节点参与共识或担任主节点的概率, 设计了分级奖惩和

抗“寡头”的动态信誉模型。相对于引言部分中方向 1 的研究工作, 本文提出的信誉模型通过信誉排序动态选取共识节点, 避免了静态选取带来的固化和不公平问题。同时进一步细化奖惩机制, 针对不同类型节点承担的功能不同, 给予不同程度的信誉奖惩, 加入惩罚因子加快对连续作恶节点的剔除, 并设计了信誉恢复机制防止“寡头”节点产生。下面是本文信誉模型的具体内容。

(1) 节点类型。在 IW-PBFT 共识算法中将节点分为三种类型: 主节点、共识节点、备份节点, 各类型节点权限如表 2 所示。主节点和共识节点可参与共识、影响力评估和权重聚合签名, 备份节点不参与共识, 但能够对聚合签名进行验证并同步共识结果。同时为保证影响力评估的公平性和正确性, 备份节点将监督影响力评估过程, 并对评估结果进行检验。

表 2 节点权限

Table 2 Node authority

节点类型	权限			
	参与共识	参与影响力评估	参与权重聚合签名	监督影响力评估
主节点	✓	✓	✓	
共识节点	✓	✓	✓	
备份节点			✓	✓

(2) 信誉排序: 在每轮共识前, 将对节点信誉值进行排序, 信誉排名前 N–f 的节点为共识节点, 其余为备份节点。在本文中对信誉值 C 设置了 4 个等级, 分别为 $C_{\min}$、C_{init}、C_{N-f}、$C_{\max}$, 大小关系为: $C_{\min}<C_{\text{init}}<C_{N-f}<C_{\max}$。其中, $C_{\max}$ 为节点能够达到的最大信誉值, 本文设置为 100, C_{N-f} 为当前轮次信誉排序的阈值, 是一个动态值, 大于该值的节点认为是相对可信的。C_{init} 为节点加入系统的初始值, 本文设置为 50, $C_{\min}$ 为节点信誉最小值, 本文设置为 30。

(3) 信誉奖惩规则: 为了提升节点参与共识的积极性, 降低恶意节点对共识过程的影响, 根据节点在共识过程中的行为和所属类型进行信誉奖惩。对于节点行为的判断, 如果该节点积极参与共识和维护区块链数据, 并产生有效区块, 则认为该节点是诚实节点, 给予奖励。若在共识过程中出现作恶行为, 如恶意篡改区块数据、发送虚假消息等破坏共识的行为, 将被认为是恶意节点, 对其进行惩罚。在每一轮共识完成后对节点进行信誉更新, 信誉更新计算公式为

$$C_{\text{new}}=C_{\text{old}}+C_{\text{reward}}-C_{\text{punish}} \tag{1}$$

其中, C_{new} 表示节点更新后的信誉值, C_{old} 表示节点的历史信誉值, C_{reward} 和 C_{punish} 分别表示对节点的信誉奖励和惩罚。

1) 信誉奖励 C_{reward}。信誉奖励用于激励在共识过程中表现好的节点。信誉奖励的计算公式为

$$C_{\text{reward}}=\begin{cases}\alpha\dfrac{C_{\text{type}}}{e}, & \text{if}\quad C_{\text{old}}\ge C_{N-f}\\ \dfrac{\alpha}{2e}, & \text{if}\quad C_{\text{old}}<C_{N-f}\end{cases} \tag{2}$$

信誉奖励公式依据节点类型设置了三种不同程度的激励, 每轮增长均控制在 0~1。首先通过节点历史信誉值 C_{old} 和当前轮次信誉排序阈值 C_{N-f} 判断是否属于共识节点, 并在共识节点激励函数中加入节点类型属性 C_{type} 进一步对主节点进行区分。在式(2)中, α 为共识参与因子, 若节点因非拜占庭原因未能参与本轮共识则该值为 0, 不获得本轮奖励。C_{type} 表示节点的类型, 主节点在共识过程中负责转发客户端的所有消息并领导整个共识, 承担的责任更重, 因此将主节点的 C_{type} 值设为 2, 共识节点设为 1。备份节点在正确检验影响力评估结果和同步共识结果后, 将给予少量奖励从而提高备份节点积极性。

2) 信誉惩罚 C_{punish}。信誉惩罚用于惩戒并剔除在共识过程中出现恶意行为的节点。信誉惩罚的计算公式为

$$C_{\text{punish}}=\begin{cases}\beta\dfrac{10C_{\text{type}}}{e}, & \text{if}\quad C_{\text{old}}\ge C_{N-f}\\ C_{\text{old}}-C_{\min}, & \text{if}\quad C_{\text{old}}<C_{N-f}\end{cases} \tag{3}$$

信誉惩罚公式与奖励公式一样设置了三种不同程度的惩罚。在式(3)中, 为了更快将恶意节点剔除, 增加了惩罚因子 β, 初始值为 1 并且将随着节点作恶次数增加进行自增。主节点获得的激励更多因此当主节点作恶时受的惩罚也更重, 由于主节点的 C_{type} 值为 2, 共识节点为 1, 根据共识节点惩罚函数, 当主节点作恶时将扣除 20 轮奖励, 共识节点作恶时扣除 10 轮奖励。若备份节点出现作恶行为, 根据备份节点惩罚函数将获得惩罚量 $C_{\text{old}}-C_{\min}$, 通过信誉更新计算公式结算后, 其信誉值置为 $C_{\min}$, 将无法再参与共识。

(4)信誉恢复机制: 为了防止部分节点信誉达到最大值 $C_{\max}$ 后始终处于高位不变, 长期参与共识成为“寡头”节点, 设置了信誉恢复机制。在执行信誉恢复机制时, 将信誉值达到 $C_{\max}$ 的节点恢复到区间

$[C_{\text{init}}, C_{N-f}]$中的随机值, 并为其信誉点数属性 C_{count} 加 1, 该属性作为节点达到信誉最大值的证明, 在信誉值相同的情况下, 优先考虑点数大的节点参与共识。通过信誉恢复机制, 避免了“寡头”节点的产生, 同时提高了低信誉值节点的共识积极性。

3.3 主节点选取

主节点在共识过程中承担重要作用, 是客户端与副本节点之间的枢纽, 来自客户端的所有请求都将由主节点进行转发, 并引导副本节点进行共识。因此, 主节点的组织能力和影响能力关系到共识能否稳定进行。目前, 大多数研究都是基于信誉属性选取主节点, 具有一定的片面性, 因为信誉高的节点不一定能够承担组织并领导共识的工作, 其影响力可能较低并不被多数节点认可。因此, 本文提出一种基于节点影响力的主节点选择方法, 目的在于辨别共识节点网络中的关键节点, 并提升关键节点当选主节点概率。主要包含两部分: 共识节点影响力评估和基于节点影响力的可验证随机函数。

3.3.1 共识节点影响力评估

为了优化 1.2 节中 K-Shell 算法存在的问题, 使其划分结果更细粒度, 本文提出一种结合整体和局部结构的 K-Shell 算法(GLK-Shell)。在 GLK-Shell 中, 整体结构的影响上继续通过 K-Shell 算法进行划分, 在局部结构的影响中将结合迭代次数、一阶和二阶邻居因素, 最终通过整体影响和局部影响得到节点的综合影响力。

GLK-Shell 的具体步骤如下。

步骤 1: 使用 K-Shell 算法进行整体结构影响力分析, 得到各节点的 Ks 值, 并记录各节点被剔除时的迭代次数 D。

步骤 2: 对各 Shell 层进行局部结构影响力分析。计算各节点的一阶邻居总数 F_{sum} 和二阶邻居总数 S_{sum}, 通过式(4)计算得到各节点的传播系数 γ。

$$\gamma = \frac{F_{\text{sum}}}{F_{\text{sum}} + S_{\text{sum}}} \tag{4}$$

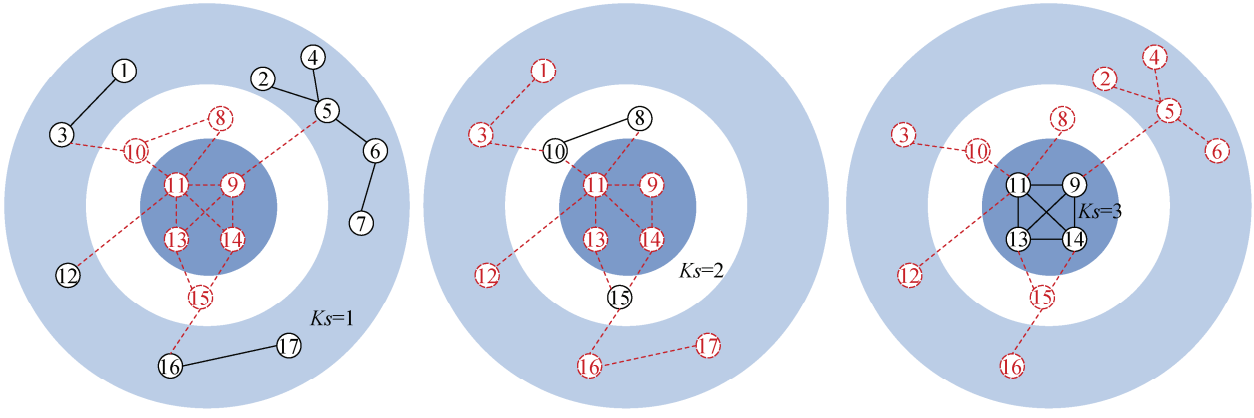
步骤 3: 根据步骤 1 得到的 D 计算各节点的迭代权重 ω, 计算公式如式(5)所示, 其中 $D_{\max}$ 为最大迭代次数。

$$\omega = \frac{D}{D_{\max}} \tag{5}$$

步骤 4: 根据步骤 1 得到的节点 K-Shell 值 Ks, 步骤 2 得到的节点传播系数 γ 和步骤 3 得到的节点迭代权重 ω, 通过式(6)计算得到节点的综合影响力 I, 其中 F_{sum} 和 S_{sum} 分别为节点的一阶和二阶邻居总数。

$$I = \omega F_{\text{sum}} + \gamma S_{\text{sum}} + Ks \tag{6}$$

GLK-Shell 的分解过程如图 5 所示。首先, 根据 K-Shell 算法的划分结果提取各 Shell 层的节点及其连边, 图中用黑色实线表示, 如图 5(c)中的 3-Shell 层节点 9、11、13、14, 以及它们的连边。其次, 各节点使用广度优先搜索遍历原始图, 获得各节点的一阶和二阶邻居节点。最后, 连接非同一 Shell 层的一阶和二阶邻居节点, 得到各 Shell 层的局部结构图, 图中用红色虚线表示。如图 5(c)中的节点 11, 它的非同一 Shell 层的一阶邻居为节点 8、10、12, 二阶邻居为节点 3、5、15。



(a) 1-Shell局部结构 (b) 2-Shell局部结构 (c) 3-Shell局部结构

图 5 GLK-Shell 分解示例图

Figure 5 GLK-Shell decomposition example

以图 5(a)1-Shell 局部结构图的节点 5 为例说明 GLK-Shell 的计算过程。首先, 节点 5 的整体结构影响力 $Ks_5 = 1$, 迭代次数 $D_5 = 3$, 最大迭代次数 $D_{\max} = 5$。其次, 节点 5 的一阶邻居为节点 2、4、6、

9, 二阶邻居为节点 7、11、13、14, 因此 $F_{\text{sum}}(5)=4$、$S_{\text{sum}}(5)=4$, 可得到节点 5 的传播系数 $\gamma_5=0.5$, 迭代权重 $\omega_5=0.6$。最后, 计算得到节点 5 的综合影响力 $I_5=5.4$。同理可得节点 2 的综合影响力 $I_2=1.95$, 而通过 K-Shell 划分的结果是节点 2 与节点 5 影响力均为 1。可见, 改进后的 GLK-Shell 算法具有更强的影响力识别能力。

在通过信誉模型筛选出共识节点后, 共识节点将进行投票。投票的方向代表两个节点间建立的关系, 从而构建一个全局投票网络图 $G=(n,e)$, n 代表当前共识节点集合, e 代表边集合。通过 GLK-Shell 算法对共识节点网络进行影响力评估。

GLK-Shell 算法如算法 1 所示。

算法 1.GLK-Shell 算法

输入: 图 $G=(n,e)$、Ks 值、迭代次数 D

输出: 节点综合影响力 I

```
K ← 1    //初始化 Shell 层数
WHILE K-Shell.nodelist > 0 DO    //遍历各 Shell 层
    FOR each node in K-Shell.nodelist:    //计算各 Shell 层节点影响力
        F_sum and S_sum ← Query.G(n,e)    //查询一阶和二阶邻居数量
        γ ← Calculate(F_sum, S_sum)    //计算传播系数
        ω ← D / D_max    //计算迭代权重
        I ← ωF_sum + γS_sum + Ks    //计算节点综合影响力
    END FOR
    K++    //下一 Shell 层
END WHILE
RETURN I    //返回节点综合影响力
```

3.3.2 基于节点影响力的可验证随机函数

为了实现对不同影响力权重节点的公平抽取, 并提升关键节点当选主节点的概率, 本文设计了基于节点影响力的可验证随机函数(I-VRF)。主要包含密钥生成函数、随机数和证明生成函数、验证函数、影响力抽签算法和抽签验证函数五部分。

(1) 密钥生成函数

$$\text{KeyGen}(E,O,r)\rightarrow(\text{pk},\text{sk}) \tag{7}$$

节点通过椭圆曲线密码学方法生成公私钥 pk, sk。选取椭圆曲线 E 以及曲线上基点 O, 通过随机数生成器生成随机整数 r 作为节点私钥, 则公钥为 $\text{pk}=r\times O$。

(2) 随机数和证明生成函数

$$\text{Testify}(\text{sk},m)\rightarrow(R,\text{proof}) \tag{8}$$

通过节点的私钥 sk 和消息 m 生成随机数 R 和证明 proof, 随机数 R 均匀分布在 $0\sim2^{\text{len}}$, len 是随机数的位长, 证明 proof 能够用来验证随机数 R 的合法性。

(3) 验证函数

$$\text{Verify}(\text{pk},m,R,\text{proof})\rightarrow\text{false}/\text{true} \tag{9}$$

输入待验证随机数 R 和它的证明 proof 以及节点公钥 pk 和消息 m, 随机数合法验证结果为 true, 否则为 false。

(4) 影响力抽签算法

根据每个节点影响力权重大小进行抽签, 使节点被抽中概率与影响力大小呈正相关。将本轮共识节点综合影响力总和 I_{sum} 按单位影响力 t 分为 $N=\frac{I_{\text{sum}}}{t}$ 份, 若节点 i 影响力为 I_i, 则该节点拥有 $n_i=\frac{I_i}{t}$ 份单位影响力。每份单位影响力被抽中概率为 $p=\frac{t}{I_{\text{sum}}}$, 则未抽中概率为 $1-p$, 服从二项分布 $B(N,p)$。

通过验证函数 Verify 验证随机数 R 的合法性后, 计算 $e=\frac{R}{2^{\text{len}}}$, 则 e 在[0,1]均匀分布, 对应二项分布累计概率范围。各节点根据自身影响力获得的份额 n_i 以概率 p 进行 n_i 次伯努利实验, 按式(10)计算成功 $k\in[0,n_i]$ 次的累计概率。

$$P(X\le k)=\sum_{i=0}^{k}C_n^i p^i(1-p)^{n-i} \tag{10}$$

$$I=\left[\sum_{i=0}^{j}C_n^i p^i(1-p)^{n-i},\sum_{i=0}^{j+1}C_n^i p^i(1-p)^{n-i}\right] \tag{11}$$

观察各节点随机数 e 落在区间 I 时对应的 $j(j\le k)$ 值, 区间 I 为公式(11)所示。满足条件的 j 为节点 i 的抽签结果, 此时有以下几种情况:

1) 各节点 j 值均为 0, 抽签失败。

2) 只有一个节点 j 值大于 0, 该节点抽签成功, 成为主节点。

3) 多个节点 j 值大于 0, 多个节点抽签成功, 选择 j 值最大的节点成为主节点(该节点被抽中最多份额数的单位影响力)。

影响力抽签算法如算法 2 所示。

算法 2.影响力抽签算法

输入: 影响力总和 I_{sum}、节点影响力 I_i、公钥 pk、消息

m、随机数 R、证明 proof
输出: 节点对应 j 值

IF Verify(pk, m, R, proof) == false **DO** //判断随机数 R 的合法性
RETURN 0
ELSE
$n_i \leftarrow I_i / t$ //计算各节点获得单位影响力份额数量
$p \leftarrow t / I_{\text{sum}}$ //单位影响力被抽中概率
$e \leftarrow R / 2^{\text{len}}$ $j \leftarrow 0$ //将随机数 R 映射到[0,1], 初始化 j 值
WHILE $e < \sum_{i=0}^{j} C_n^i p^i (1-p)^{n-i}$ **DO** //判断是否抽中
$j++$
END WHILE
RETURN j //返回节点抽中份额数
END IF

(5) 抽签验证函数

$$\text{VerifySelect}(\text{pk}, R, m, \text{proof}, j) \rightarrow \text{false} / \text{true} \quad (12)$$

验证函数首先会检验被抽中节点随机数的合法性, 仅当随机数合法时, 验证函数通过相同的方法获得节点被抽中份额数 j'。若 $j = j'$,则说明是有效抽签结果, 返回 true, 否则返回 false。

3.4 IW-PBFT 共识流程设计

3.4.1 权重聚合签名

在传统的聚合签名[20]中, 所有参与者被视为同等地位, 有着相同的权重, 但在实际中不同的参与方可能具有不同的可信度或重要性, 而聚合签名并不能够反映出这种重要性差异。同时, 传统的聚合签名缺少灵活性, 不能够根据实际场景动态调整不同参与者的权重。

因此, 为了提高聚合签名的灵活性, 反映不同节点重要性差异, 本文设计了权重聚合签名。通过节点影响力大小为签名节点分配权重, 判断权重是否达到阈值来减少所需的签名次数。主要包含节点权重分配、密钥生成、权重聚合签名和验证签名及权重四部分。

(1) 节点权重分配

在评估完节点的综合影响力后, 将对节点进行影响力排序, 节点排名越高, 获得的权重越大。当共识节点数量为 N 时, 节点通过式(13)获得自身的签名权重 W, 其中, k 为节点的影响力排名, $k \in [1, N]$, $W \in (0, 1/3]$。

$$W = \frac{1}{k+2} \quad (13)$$

(2) 密钥生成

假设 g_1 和 g_2 分别为乘法循环群 G_1 和 G_2 上的生成元, 存在双线性映射 $e: G_1 \times G_2 \rightarrow G_3$, 同时对消息 m, 存在哈希函数 $h: h(m) \in G_2$。则签名节点选取随机数 r 作为私钥, 公钥为 $\text{pk} = g_1^r$。

(3)权重聚合签名

签名节点使用私钥对消息 m 进行签名, 得到签名 $\text{Sign} = h(m)^r$, 且 $S \in G_2$。验证者可通过签名节点的公钥验证等式 $e(g_1, \text{Sign}) = e(\text{pk}, h(m))$ 是否成立来检验签名 Sign 的合法性。在验证通过后, 节点将自身签名、公钥和权重信息发送给聚合节点, 聚合节点通过式(14)判断权重是否达到阈值进行聚合签名操作。在 PBFT 中需收集总权重 2/3 的合法签名信息才能进行下一阶段, 为了进一步减少共识过程中的通信开销和签名量, 在本文中将权重阈值设定为总权重的 1/2。

$$\sum_{i=1}^{n} W_i \geq W_{\text{threshold}} \quad (14)$$

在通过权重阈值判断后, 聚合节点将这些签名累乘得到一个聚合签名 AggSign, 同时生成累计权重证明 SProof 和节点公钥列表 PkList, 并构造结构体 $\sigma = \{\text{AggSign}, \text{SProof}, \text{PkList}\}$ 为后续签名验证做准备。

权重聚合签名过程如图 6 所示, 节点 1、节点 2、节点 3 为参与节点。节点使用私钥对消息进行签名得到各自签名 Sign_i, 在通过合法性验证后, 各节点将自身公钥、权重和签名发送给聚合节点。当收到

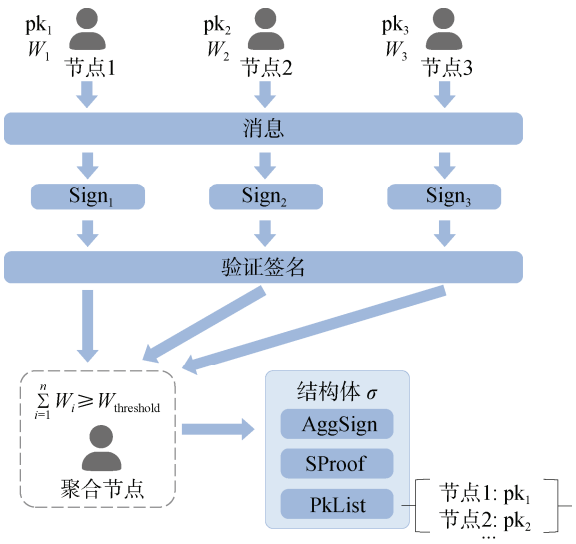


图 6 权重聚合签名过程

Figure 6 Weighted aggregation signature process

的累计权重达到系统设定的阈值后, 聚合节点将构造结构体 σ, 并将生成的聚合签名 AggSign、累计权重证明 SProof 和节点公钥列表 PkList 存入结构体中。

(4) 验证签名及权重

在此阶段中, 各节点均可通过结构体 σ 来验证聚合签名 AggSign 的正确性。从聚合节点接收到结构体 σ 后, 仅当式(15)成立和累计权重证明 SProof 是合法时验证通过, 签名有效。

$$e(g_1, \text{AggSign}) = \prod_{i=1}^{N} e(\text{pk}_i, h(m)) \tag{15}$$

3.4.2 共识流程

IW-PBFT 共识流程共分为 6 个阶段: 请求、预准备、准备、权重聚合、验证和回复。具体共识流程如图 7 所示, 节点 0 为主节点, 节点 1、2、3 为共识节点, 节点 4 为备份节点, 其中节点 3 是恶意节点。

(1) 请求阶段: 客户端向主节点 0 发送附加自身签名 Sign_c 的请求消息 $<\text{REQUEST}, o, t, c>\text{Sign}_c$, 其中, o 为具体的操作, t 为时间戳, c 表示客户端编号。

(2) 预准备阶段: 主节点 0 检查请求消息的正确性, 验证通过后在当前视图 v 中为其分配序列号 s, 并广播预准备消息 $<<\text{PRE}-\text{PREPARE}, v, s, d>, \text{Sign}_0, m>$ 给共识节点 1、2、3。其中, d 为消息 m 的摘要, Sign_0 是主节点 0 对预准备消息的签名。

(3) 准备阶段: 预准备消息验证通过后, 共识节点间相互广播准备消息 $<<\text{PREPARE}, v, s, d>, N_{\text{id}}, \text{Sign}_i>$, 其中 N_{id} 为共识节点编号, Sign_i 是共识节点签名。图中共识节点 1、2、3 相互广播准备消息, 因恶意节点 3 发送错误消息干扰共识, 仅当共识节点至少有 $2f$ 个一致的准备消息, 主节点为 $2f+1$ 个时, 进入权重聚合阶段。

(4) 权重聚合阶段: 共识节点 1、2、3 将签名、公钥和权重信息追加到消息中, 主节点 0 等待并收集共识节点的有效签名。此阶段恶意节点 3 可发送虚假消息但签名是合法的, 因此需判断消息的一致性, 仅当主节点 0 收到消息的累计权重超过聚合阈值并且这些消息是一致时, 将生成聚合签名 AggSign, 并构造结构体 σ 追加到消息中发送给其他节点。

(5) 验证阶段: 共识节点 1、2、3 和备份节点 4 收到消息后, 通过消息中的结构体 σ 获取参与聚合的节点公钥列表 PkList 和累计权重证明 SProof, 利用公式(15)对聚合签名 AggSign 进行合法性验证。当签名验证正确并且累计权重证明是合法时, 同时请求内容与准备消息一致, 节点执行客户端请求操作。

(6) 回复阶段: 主节点 0 和共识节点 1、2 完成客户端请求操作后, 发送回复消息 $<\text{REPLY}, v, t, c, \text{Sign}_i, r>$, r 为请求操作的执行结果,客户端收到 $f+1$ 个一致的执行结果后, 认为请求已经得到执行, 本轮共识完成。

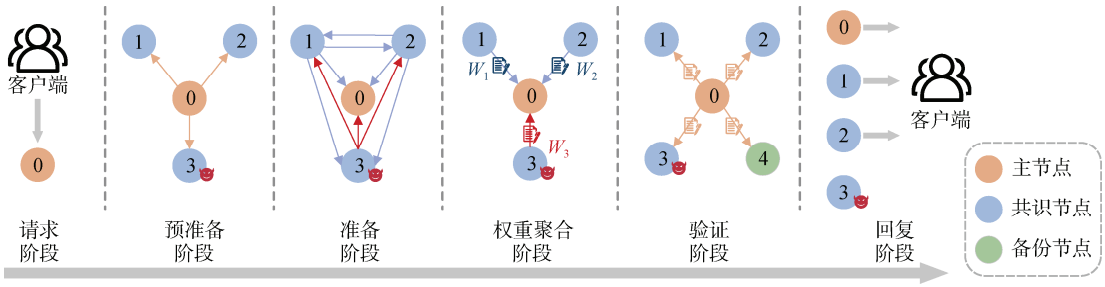


图 7 IW-PBFT 共识流程

Figure 7 IW-PBFT consensus process

在回复阶段中, 多数诚实的节点将及时响应主节点请求并回复客户端, 少部分节点因网络原因未能及时同步请求。在共识完成后, 共识节点和备份节点可以相互交换本地区块的 Merkle 根和每个区块的哈希值, 若 Merkle 根或区块哈希不匹配, 节点可根据区块顺序请求最新的数据进行同步, 确保数据的一致性和完整性。

4 算法分析

4.1 安全性分析

本文提出的 IW-PBFT 共识算法是基于 PBFT 算法改进的, 在容错能力上与 PBFT 算法相当, 可容忍不超过共识节点总数 1/3 的恶意节点。为了防止主节点可预测问题, 采用了基于节点影响力的可验证随机函数抽签算法来选取主节点。各个节点通过私钥 sk 和消息 m 生成唯一的随机数 R 和证明 proof, 并在可验证随机函数的基础上设计了影响力权重抽签, 根据节点影响力分配不同的单位影响力份额, 参与抽签后根据被抽中单位影响力份额数 j 判断抽签结果。因此, 对手在不知道节点私钥和影响力权重的情况下, 无法通过伪造随机数和证明或猜测节点被抽中份额数来预测主节点, 能够有效地防止主节点可

预测问题。

定理 1(交易确定性): IW-PBFT 共识算法具有交易确定性。当恶意节点数量在容错范围内时, 在同一轮次中, 对同一个主节点的交易请求, 所有诚实节点不会输出不同的结果。

证明: 反证法。前提条件: 系统中存在 n 个共识节点, 其中最多有 f 个节点是恶意的, 数量关系满足 $n \geq 3f+1$。假设 IW-PBFT 共识算法不能保证交易确定性, 即在同一轮次中存在两个诚实节点对交易产生不同的输出。对于诚实节点 i 和 j, 交易输出结果不同的条件为: 准备阶段和验证阶段对请求消息 m 的摘要 d 不一致, 即节点 i 认为摘要为 d_1, 节点 j 认为摘要为 d_2。能达成上述条件的仅有主节点在预准备阶段中向 $\frac{n-f}{2}$ 个诚实节点发送消息 $\text{msg}_1 = <\text{PRE}-\text{PREPARE}, d_1>$, 向另外 $\frac{n-f}{2}$ 个诚实节点发送消息 $\text{msg}_2 = <\text{PRE}-\text{PREPARE}, d_2>$。这样在准备阶段中 $\frac{n-f}{2}$ 个诚实节点将对摘要 d_1 达成一致, 另外 $\frac{n-f}{2}$ 个诚实节点对摘要 d_2 达成一致, 而进入权重聚合阶段的必要条件为主节点至少有 $n-f$ 个一致的来自不同节点的准备消息, 即 $\frac{n-f}{2}+f \geq n-f$, 解得 $n \leq 3f$, 与要求 $n \geq 3f+1$ 不符, 并且主节点恶意行为将触发视图更换。因此假设不成立, IW-PBFT 共识算法具有交易确定性。

定理 2(区块一致性): IW-PBFT 共识算法具有区块一致性。在容错范围内, 若某个诚实节点在一轮共识结束后成功提交区块 Block_i, 则所有诚实节点的输出结果均为 Block_i。

证明: 假设诚实节点 i 在一轮共识后输出区块 Block_i, 则该区块一定完整经过了共识流程, 由定理 1 可知, IW-PBFT 共识算法具有交易确定性, 所有诚实节点对区块 Block_i 内的交易均能达成一致, 因此所有诚实节点输出结果均为 Block_i, IW-PBFT 共识算法具有区块一致性。

定理 3(顺序一致性): IW-PBFT 共识算法具有顺序一致性。在容错范围内, 若某个诚实节点输出的区块顺序为 $\{\text{Block}_1, \text{Block}_2, \cdots, \text{Block}_n\}$, 那么所有诚实节点的区块顺序均为 $\{\text{Block}_1, \text{Block}_2, \cdots, \text{Block}_j\}$, 其中 $j \leq n$。

证明: 由定理 2 可知, IW-PBFT 共识算法具有区块一致性, 所有的诚实节点都将输出相同的区块, 在共识流程中, 区块将依次输出。因此所有诚实节点最终输出的区块顺序均能一致, IW-PBFT 共识算法具有顺序一致性。

4.2 可行性分析

在一个典型的基于区块链的电子病历共享系统中, 参与方包括医院、监管机构、诊所、药店和保险公司, 共同维护和管理患者的医疗数据。医院和监管机构具有较高的重要性, 而诊所和药店重要性相对较低。若使用传统 PBFT 共识算法存在以下问题: (1)缺少信誉机制, 低信誉的参与方参与共识可能会由于自身漏洞或恶意行为增加医疗数据被篡改、泄露的风险。(2) 各参与方轮流担任主节点存在可预测风险, 若低信誉的诊所或药房成为主节点, 将会存在较大的安全隐患。(3) 通信开销大, 大量参与方的加入会导致性能下降, 对高效、低延迟的电子病历管理和共享系统不利。

而本文提出的 IW-PBFT 共识算法首先通过信誉模型筛选节点进行共识, 使得高信誉参与方在共识中占据主导地位。其次, 提出的主节点选择方法在保障主节点选择随机性的同时, 增大影响力大的关键参与方当选概率, 例如高信誉、高级别的医院和监管机构。最后, 提出的权重聚合签名方案符合实际应用场景中各参与方可能具有不同的可信度或重要性的需求, 例如不同级别的医院、医院与诊所之间。综上所述, IW-PBFT 共识算法具有较好的可行性和广阔的应用前景。

4.3 签名量和通信开销分析

(1) 签名量分析

在 1.1 节中分析了 PBFT 算法中确认阶段所需的签名和验证签名总次数为 $2N^2-2N$。对于聚合签名, 在确认阶段中, 需收集节点总数 2/3 的签名才能生成聚合签名, 则最少需要的签名次数为 $\frac{2}{3}N$, 每个节点将对聚合签名进行验证, 验证签名次数为 N。因此, 聚合签名在确认阶段中所需的最小签名量为 $\frac{5}{3}N$。可以看出, 聚合签名方案所需的签名量远远小于原 PBFT 算法。

本文权重聚合签名主要通过给予节点不同的权重, 快速达到聚合签名所需的条件。假设共识节点总数为 N, 节点根据影响力排名获得自身权重, 则系统中总权重 W_{sum} 为

$$W_{\text{sum}}=\sum_{i=1}^{n}\frac{1}{i+2} \tag{16}$$

权重阈值设置为总权重的一半, 因此, 对于权重聚合签名, 最少需要的签名次数为满足不等式(17)所对应的最小 t 值, 验证签名次数与聚合签名相同为 N。

$$\sum_{i=1}^{t}\frac{1}{i+2}\geq\frac{W_{\text{sum}}}{2} \tag{17}$$

权重聚合签名与聚合签名方案最小签名量对比如图 8 所示。从图 8 可以看出, 随着节点数量增加, 两种方案所需的最小签名量都呈现上升趋势, 但权重聚合签名方案增长趋势较缓, 且签名量始终低于聚合签名方案, 相比于聚合签名方案有着更低的聚合门槛。

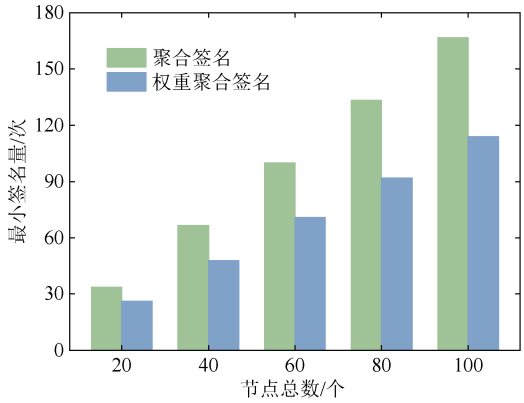


图 8 最小签名量对比

Figure 8 Minimum signature volume comparison

(2) 通信开销分析

通信开销是指在共识过程中节点间产生的通信次数总和。IW-PBFT 算法通过减少参与共识的节点数量和权重聚合签名减少了共识过程的通信开销。设节点总数为 N, 在节点数量较多且经历几轮共识后, 共识节点个数近似为总节点数的 2/3。具体来说, 各个阶段通信开销分别为: 请求阶段通信次数为 1; 预准备阶段通信次数为 $\frac{2}{3}N-1$; 准备阶段通信次数为 $\left(\frac{2}{3}N-1\right)^2$; 聚合阶段通信次数为 $\left(\frac{2}{3}N-1\right)$; 验证阶段通信次数为 $(N-1)$;回复阶段通信次数为 $\frac{2}{3}N$。因此 IW-PBFT 共识算法正常情况下的通信开销 T_I 是这六阶段总和:

$$\begin{aligned}T_I&=1+2\left(\frac{2}{3}N-1\right)+\left(\frac{2}{3}N-1\right)^2+(N-1)+\frac{2}{3}N\\&=\frac{4}{9}N^2-\frac{5}{3}N-1\end{aligned} \tag{18}$$

同时分析了 PBFT、文献[15]提出的 C-PBFT 以及文献[16]提出的 tPBFT 算法从请求阶段到回复阶段的通信开销, 如表 3 所示。进一步得到了节点数量变化时各算法的通信开销对比图, 如图 9 所示。

表 3 通信开销对比

Table 3 Comparison of communication overheads

算法名称	通信开销
PBFT	$2N^2-N+1$
C-PBFT	$\frac{8}{9}N^2-\frac{2}{3}N+1$
tPBFT	$\frac{1}{2}N^2-\frac{1}{2}N+1$
IW-PBFT	$\frac{4}{9}N^2-\frac{5}{3}N-1$

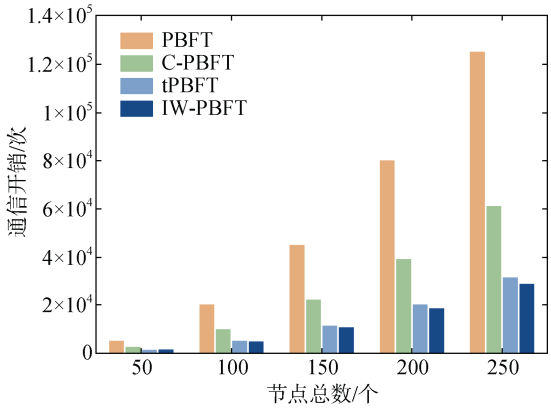


图 9 各算法通信开销对比

Figure 9 Comparison of communication overheads for each algorithm

从图 9 可以看出, 随着节点数量增加, PBFT 通信开销增长最快, 而 IW-PBFT 增长最慢, 并且始终低于 C-PBFT 和 tPBFT。tPBFT 由于限制了一半节点参与共识, 大幅度降低了通信开销, 但由于并未对共识流程进行改进, 其通信开销始终高于 IW-PBFT。总体而言, 改进后的 IW-PBFT 算法通信开销更低, 能够有效降低通信开销。

5 实验及结果分析

为了验证 IW-PBFT 共识算法的有效性, 本实验在采用 Intel(R) Core(TM) i7-13700HX 2.10GHz 处理器、16GB 内存的 Windows 11 系统下进行, 软件环境为 Go 1.18 和 IntelliJ IDEA, 采用多线程模型, 每个节点作为独立线程运行, 实验中开启多个线程以模拟分布式系统中节点的并发活动, 通过日志记录系统监测共识过程中的性能指标, 实验数据用 Origin 进行绘制。将从节点信誉变化、恶意节点剔除、主

节点选择正确性、吞吐量和交易时延五个部分进行实验分析。

5.1 节点信誉变化

为了验证本文信誉模型是否能够有效惩罚恶意节点以及区分不同类型节点, 在实验中设置了 4 种不同属性的节点, 分别为: 诚实主节点(Node1)、诚实共识节点(Node2)、中途作恶节点(Node3)和诚实备份节点(Node4)。进行 20 轮共识, 并设置第 17 轮为作恶轮次, 作恶节点将在该轮发送篡改后的信息, 诚实节点发送原始消息, 通过信誉模型更新节点信誉, 节点信誉变化如图 10 所示。

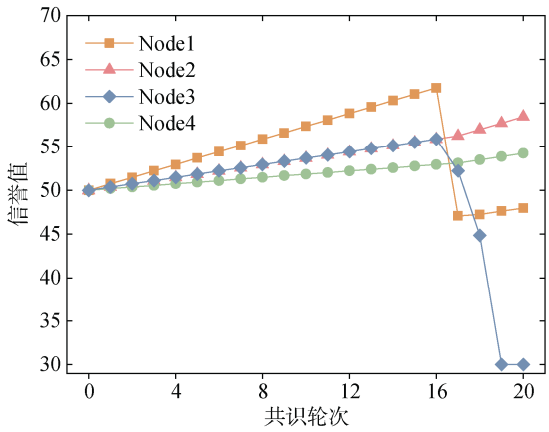


图 10 节点信誉变化

Figure 10 Node reputation changes

从图 10 可以看出, 对于节点 1、节点 2 和节点 4 具有不同的信誉增长趋势, 信誉模型能够区分不同类型节点。在第 17 轮作恶轮次中, 节点 1 和节点 3 将进行作恶, 节点 1 因其是主节点, 将被扣除 20 轮的奖励, 而节点 3 是共识节点将被扣除 10 轮获得的奖励。经过新一轮信誉排序, 节点 1 被划为备份节点, 节点 4 成为共识节点, 节点 2 成为主节点。而节点 3 将连续作恶, 惩罚因子 β 将进行自增, 惩罚力度加大, 在第 18 轮节点 3 将被扣除 20 轮获得的奖励, 经过信誉排序节点 3 成为备份节点, 节点 1 成为共识节点, 在第 19 轮中信誉模型将作恶的备份节点 3 信誉值降至 30。实验表明, 本文信誉模型能够区分不同类型节点并给予不同强度的奖惩, 同时能够动态地对节点进行类型划分。

5.2 恶意节点剔除

为了检验信誉模型识别并剔除恶意节点的能力。实验中设置了 6 个恶意节点, 这 6 个恶意节点刚开始都是诚实节点, 随着共识进行信誉值趋于稳定后, 将随机发送篡改后的消息干扰共识, 每轮共识结束后, 记录共识节点中恶意节点的数量, 观察在 10 轮共识中 PBFT、C-PBFT 和 IW-PBFT 共识算法恶意节点数量随共识轮次增加的变化情况, 实验结果如图 11 所示。

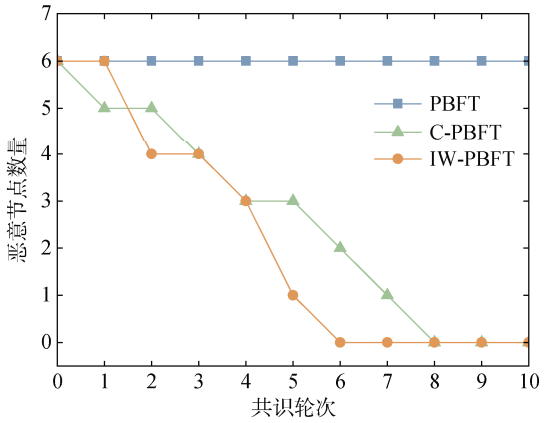


图 11 恶意节点数量对比

Figure 11 Comparison of the number of malicious nodes

由实验结果可以看出, PBFT 算法由于缺少奖惩机制, 恶意节点将始终存在, 严重影响系统安全性, 而 C-PBFT 和 IW-PBFT 中恶意节点数量呈现下降趋势, 能够有效减少恶意节点数量。由于 IW-PBFT 共识算法中设置了惩罚因子, 恶意节点连续作恶时将比 C-PBFT 算法更加快速剔除, 因此 IW-PBFT 共识算法能够有效地识别恶意节点, 并快速剔除, 提升了系统安全性。

5.3 主节点选择正确性

为了验证 GLK-Shell 算法和 I-VRF 算法的有效性, 本文分别对 GLK-Shell 和 I-VRF 进行了影响力评估测试和节点抽取测试。实验中设置了 15 个共识节点进行模拟投票并记录各节点的投票结果, 构建全局投票图, 分别使用 K-Shell 算法和 GLK-Shell 算法进行影响力评估, 结果如图 12 所示。

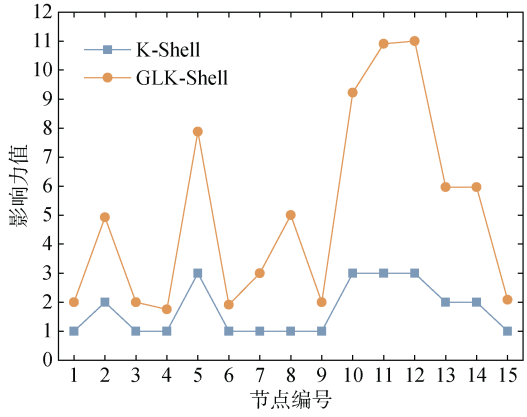


图 12 节点影响力评估对比

Figure 12 Comparison of node influence evaluation

从图 12 可以看出，与 K-Shell 算法相比，GLK-Shell 算法的划分结果更细粒度，K-Shell 算法将 15 个节点仅划分为 3 个等级，而 GLK-Shell 算法进一步将这些节点划分为更多等级。同时，GLK-Shell 算法对同层节点区分度更高，例如 K-Shell 算法将节点 5、10、11、12 划为 3-Shell 层，具有相同的影响力值，不能够很好地区分这些节点影响力，而 GLK-Shell 算法通过局部影响力分析进一步对节点区分，将这 4 个节点划分为 4 个等级，具有不同的影响力值。因此，GLK-Shell 算法有着更好的影响力识别能力。

将 15 个节点按影响力大小重新编号，分别利用 VRF 和 I-VRF 进行 200 次有效抽取，记录各节点抽中次数，实验结果如图 13 所示。

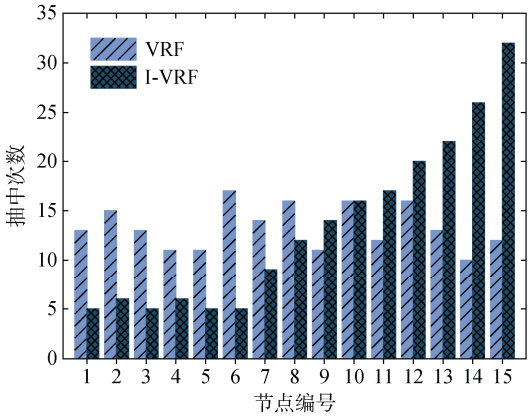


图 13　抽取结果分布

Figure 13　Distribution of selection results

从图 13 可以看出，VRF 是均匀随机抽取，所有节点被抽中概率总体一致，但均匀随机抽取容易抽中影响力低的非关键节点。例如影响力低的节点 2 和节点 6，通过 VRF 被抽中的次数比影响力高的关键节点 14 和节点 15 更多，所以 VRF 并不能区分不同影响力节点被抽中概率，无法提升关键节点当选概率。而从 I-VRF 抽取结果可以看出，节点被抽取概率总体上与自身影响力大小呈正相关，能够区分不同影响力节点被抽中概率。因此，I-VRF 在保证主节点随机选取的同时，又提升了关键节点当选概率，增强了共识的稳定性。

5.4　吞吐量

吞吐量是指系统在单位时间内完成交易的数量，反映了系统处理请求的能力，是评估算法性能的重要指标，吞吐量越高算法性能越好，一般用 TPS (Transaction Per Second)表示，吞吐量的计算公式如式(19)所示。

$$\mathrm{TPS}=\frac{\mathrm{Transactions}}{\Delta t} \tag{19}$$

其中，Δt 表示处理交易花费的时间，Transactions 表示处理的交易总数。为了验证 IW-PBFT 算法的高效性，本文与 PBFT 算法和 C-PBFT 算法进行吞吐量对比分析，实验中节点总数分别设置为 5、20、40、60、80、100、120，每组节点重复进行 20 次测试，测试结果取平均值。三种算法的吞吐量对比如图 14 所示。

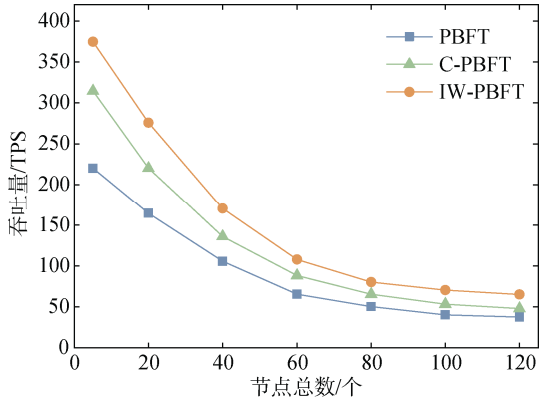


图 14　吞吐量对比

Figure 14　Comparison of troughput

从图 14 可以看出，当节点数量逐渐增多时，三种算法吞吐量都呈下降趋势，但 IW-PBFT 共识算法的吞吐量整体大于 PBFT 和 C-PBFT 共识算法。在节点数达到 100 时，三种算法趋于平缓，IW-PBFT 算法保持在 70TPS 状态，平均吞吐量与 PBFT 相比提高了 65.7%。PBFT 共识算法有着较高的通信复杂度，并且需要全部节点参与共识导致吞吐量较低，C-PBFT 共识算法减少了参与共识节点数量，但并未优化共识流程，整体吞吐量低于 IW-PBFT 算法。而 IW-PBFT 共识算法在减少共识节点数量的基础上优化了共识流程，有效提高了吞吐量。由于需要维护全局网络图和进行影响力评估，当节点数量较多时吞吐量呈下降趋势，但总体上优于 PBFT 和 C-PBFT 共识算法，有更好的吞吐量性能，能够提升系统的共识效率。

5.5　交易时延

交易时延是指从客户端提交交易请求到共识完成请求被执行所花费的时间，时延越低交易确认的速度越快，与吞吐量一样都是评估算法性能的重要指标，交易时延的计算公式如式(20)所示。

$$\mathrm{Latency}=T_{\mathrm{reply}}-T_{\mathrm{request}} \tag{20}$$

其中，T_{reply} 表示客户端收到回复消息的时间，T_{request} 表示客户端发送请求的时间。在与吞吐量测试相同的实验条件下进行交易时延测试，并进行对比分析，三种算法的交易时延对比如图 15 所示。

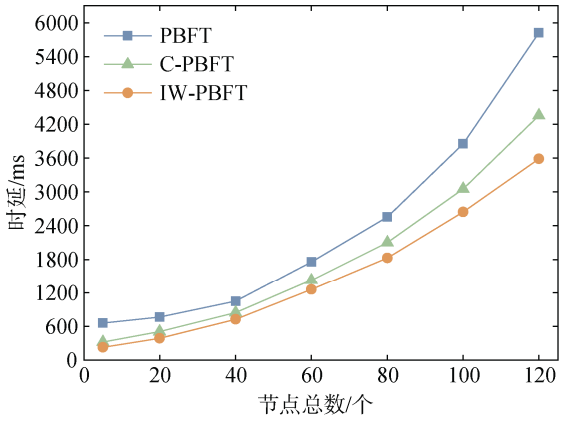


图 15　交易时延对比

Figure 15　Comparison of transaction latency

从图 15 可以看出, 随着节点数量增加, 三种共识算法的时延都呈现出增长趋势, 但 IW-PBFT 算法时延增长率更低, 相比 PBFT 算法, IW-PBFT 算法平均时延降低了 38.9%。PBFT 算法复杂的通信过程和恶意节点持续存在导致频繁的视图更换使得节点数量增加时时延增幅较大, C-PBFT 算法虽然减少了共识节点数量, 但并未解决共识过程复杂的问题。而 IW-PBFT 共识算法在共识节点数量和共识流程上都进行了改进, 有效降低了交易时延。在节点数量较多时, 由于需要额外的节点间投票以及影响力评估, 导致时延有增长趋势, 但整体上 IW-PBFT 共识算法的交易时延小于 PBFT 和 C-PBFT 共识算法, 更适用于大规模节点交易场景中。

6　结论

本文针对 PBFT 共识算法存在缺少奖惩机制、主节点选取方式可预测和通信开销大的问题, 提出了 IW-PBFT 共识算法。设计了信誉模型动态筛选出共识节点和备份节点, 依据节点类型给予不同程度的奖惩, 并设计了信誉恢复机制避免“寡头”节点产生。同时, 通过优化的 K-Shell 算法进行影响力评估, 并设计了基于节点影响力的可验证随机函数, 在保证主节点选取安全性的同时增大了关键节点当选概率。最后结合权重聚合签名优化了一致性协议的执行过程, 提高了共识效率。实验表明, 相比于 PBFT 算法及其改进算法, IW-PBFT 共识算法具有更低的通信开销、交易时延和更高的吞吐量。但 IW-PBFT 仍有不足之处, 如在节点数量较多时, 算法性能还需进一步提高。未来的研究工作将重点优化通信拓扑结构和区块存储结构, 使其更适用于实际的区块链系统中。

参考文献

[1] Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. https://bitcoin.org/en/bitcoin-paper. 2024.

[2] Jose D T, Holme J, Chakravorty A, et al. Integrating Big Data and Blockchain to Manage Energy Smart Grids—TOTEM Framework[J]. *Blockchain*: *Research and Applications*, 2022, 3(3): 100081.

[3] Fu J T, Zhang L P, Wang L X, et al. BCT: An Efficient and Fault Tolerance Blockchain Consensus Transform Mechanism for IoT[J]. *IEEE Internet of Things Journal*, 2023, 10(14): 12055-12065.

[4] Rehman M, Javed I T, Qureshi K N, et al. A Cyber Secure Medical Management System by Using Blockchain[J]. *IEEE Transactions on Computational Social Systems*, 2023, 10(4): 2123-2136.

[5] Fu X, Wang H M, Shi P C, et al. Teegraph: A Blockchain Consensus Algorithm Based on TEE and DAG for Data Sharing in IoT[J]. *Journal of Systems Architecture*, 2022, 122: 102344.

[6] King S, Nadal S. Ppcoin: peer-to-peer crypto-currency with proof-of-stake. https://decred.org/research/king2012.pdf. 2024.

[7] Li W C, Deng X H, Liu J, et al. Delegated Proof of Stake Consensus Mechanism Based on Community Discovery and Credit Incentive[J]. *Entropy*, 2023, 25(9): 1320.

[8] Xu J J, Wang W, Zeng Y, et al. Raft-PLUS: Improving Raft by Multi-Policy Based Leader Election with Unprejudiced Sorting[J]. *Symmetry*, 2022, 14(6): 1122.

[9] Zhang B C, Kong L J, Li Q Z, et al. EB-BFT: : An Elastic Batched BFT Consensus Protocol in Blockchain[J]. *Future Generation Computer Systems*, 2023, 139(C): 267-279.

[10] Gan B, Wang Y J, Wu Q W, et al. EIoT-PBFT: A Multi-Stage Consensus Algorithm for IoT Edge Computing Based on PBFT[J]. *Microprocessors and Microsystems*, 2022, 95: 104713.

[11] Liu J, Deng X H, Li W C, et al. CG-PBFT: An Efficient PBFT Algorithm Based on Credit Grouping[J]. *Journal of Cloud Computing*, 2024, 13(1): 74.

[12] Liu J, Feng W L, Zhang Y, et al. Improvement of PBFT Algorithm Based on CART[J]. *Electronics*, 2023, 12(6): 1460.

[13] Na Y H, Wen Z, Fang J, et al. A Derivative PBFT Blockchain Consensus Algorithm with Dual Primary Nodes Based on Separation of Powers-DPNPBFT[J]. *IEEE Access*, 2022, 10: 76114-76124.

[14] Wang Y, Zhong M L, Cheng T. Research on PBFT Consensus Algorithm for Grouping Based on Feature Trust[J]. *Scientific Reports*, 2022, 12: 12515.

[15] Tang S, Wang Z Q, Jiang J, et al. Improved PBFT Algorithm for High-Frequency Trading Scenarios of Alliance Blockchain[J]. *Scientific Reports*, 2022, 12: 4426.

[16] Micali S, Vadhan S, Rabin M. Verifiable Random Functions[C]. *The 40th Annual Symposium on Foundations of Computer Science*, 1999: 120.

[17] Li S Z, Xiong W Z, Deng X H, et al. Byzantine Fault-Tolerance Consensus Algorithm Based on Perfect Binary Tree Communication[J]. *Journal of Electronics & Information Technology*, 2023, 45(7): 2484-2493.
(李淑芝, 熊伟志, 邓小鸿, 等. 基于完美二叉树通信拓扑的拜占庭容错共识算法[J]. *电子与信息学报*, 2023, 45(7):

2484-2493.)

[18] Jiang W X, Wu X X, Song M Y, et al. A Scalable Byzantine Fault Tolerance Algorithm Based on a Tree Topology Network[J]. *IEEE Access*, 2023, 11: 33509-33519.

[19] Kitsak M, Gallos L K, Havlin S, et al. Identification of Influential Spreaders in Complex Networks[J]. *Nature Physics*, 2010, 6(11): 888-893.

[20] Boneh D, Gentry C, Lynn B, et al. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps[C]. *Advances in Cryptology — EUROCRYPT 2003*, 2003: 416-432.

刘力汇 于 2022 年在江西理工大学应用科学学院计算机科学与技术专业获得学士学位。现在江西理工大学网络与信息安全专业攻读硕士学位, CCF 学生会员。研究领域为区块链。Email: 6720220804@mail.jxust.edu.cn

刘勇 于 2022 年在江西理工大学应用科学学院计算机科学与技术专业获得学士学位。现在江西理工大学网络与信息安全专业攻读硕士学位。研究领域为区块链。Email: 6720220808@mail.jxust.edu.cn

张丽 于 2022 年在江西理工大学应用科学学院通信专业获得学士学位。现在江西理工大学计算机技术专业攻读硕士学位。研究领域为区块链。Email: 6720220602@mail.jxust.edu.cn

邓小鸿 于 2013 年在中南大学计算机应用技术专业获得博士学位。现任赣南科技学院信息工程学院副教授, 江西理工大学信息工程学院硕士生导师, CCF 会员。研究领域为网络信息安全、区块链。Email: deng_xh@jxust.edu.cn

石亦燃 于 2022 年江西理工大学应用科学学院通信工程专业获得学士学位。现在江西理工大学网络与信息安全专业攻读硕士学位, CCF 学生会员。研究领域为区块链。Email: 6720220801@mail.jxust.edu.cn